REMARKS

This is intended as a full and complete response to the Final Office Action dated July 14, 2004, having a shortened statutory period for response set to expire on October 14, 2004. Please reconsider the claims pending in the application for reasons discussed below.

Claims 1-26 are pending in the application.

Claims 1-6, 9-22, 25 and 26 stand rejected under 35 U.S.C. § 103(a) as being not patentable over *Blainey* (US 6,045,585) in view of *Archambault* (US 6,173,444). Applicants respectfully traverse the rejection.

Blainey relates to a method and system which "determines alias information across compilation units of a program being compiled" by manipulating an alias set. *Blainey* Col. 1 ll. 7-10. "Generally a symbol (storage location or a variable) which may share storage with another symbol is referred to as an 'alias' of, or is said to be 'aliased with', the other symbol." *Blainey* Col. 1 ll. 46-48. *Blainey* employs a shadow variable (i.e. an additional variable not part of the source code being compiled, but created by the compiler during the compilation process) to represent the set of storage locations which are possibly referenced through an indirect reference. Initially, the set may be quite large, based on a pessimistic assumption of "that any shadow in a visible compilation unit is aliased with all mapped storage locations whose addresses are explicitly taken in the visible portion of the program." *Blainey* Col. 5 ll. 58-60. The teaching of *Blainey* is a method for eliminating members from this pessimistic alias set based on an analysis that identifies variables that, in fact, *cannot* be reached by an particular indirect reference.

Archambault teaches eliminating successive, intermediate references from an alias graph so that a corresponding alias set include only "l-val" elements. *Archambault* Col 8. ll. 15-16. That is, *Archambault* teaches a method to eliminate chaining references such as A→B→C. (Read as: "A is a reference to B, which itself is a reference to C). By eliminating the intermediate link in the chain, *Archambault* discloses a method for generating an alias set that includes only the set of possible address taken variables referenced by the indirect reference (A→B). See *Archambault*, Col. 6 ll. 16-67.

Thus, in both *Blainey* and *Archambault*, the techniques serve to remove variables from alias sets that ***cannot*** be reached from a particular indirect reference, or that only play an intermediate result in resolving an actual variable reached by an indirect reference. In stark contrast, Applicants' invention serves to refine an alias set by removing variables from an address taken alias set that ***can*** be reached from an indirect reference. Specifically, independent claims 1, 11, 12, and 17 (and claims dependent therefrom) include determining whether a load of an address exists for a variable in an intermediate representation of a source code and, under certain conditions, removing the variable from an address taken alias set used with the intermediate representation. Whether a variable can be removed depends on what uses a program makes of the indirect reference in an intermediate representation of source code (i.e. a representation of computer source code after created after the first pass of a multi-pass compiler). Once removed, the indirect reference is replaced with a direct one; namely, the reference that was originally *reached* by the indirect reference.

A compiler configured to carry out the combined teachings of *Blainey* and *Archambault* would, on the other hand, leave a variable as part of the address taken alias set that are subsequently (or antecedently) removed from the alias set altogether by Applicants' invention. Put another way, any address taken variables (e.g., alias sets) and indirect references modified by Applicants' invention would be left alone by the methods disclosed by *Blainey* and *Archambault*.

The respective optimization techniques are complimentary, *i.e.*, the output of Applicants' invention may be used as an input to that of *Blainey*. Accordingly, *Blainey* in light of *Archambault*, fails to teach, show, or suggest, Applicants' claims for removing certain variables from an address taken alias set by replacing indirect references with direct references, or removing variables from an address taken alias set that **can** be reached by an indirect reference. Furthermore, *Blainey* specifically contemplates that individuals might invent new refinement techniques that work with the techniques disclosed therein:

> It will be apparent that the pessimistic [alias] sets which result from the analysis can often be somewhat refined through ... conventional data-flow analysis or refinements, or **as would occur to those of skill in the art**.

While such refinement is not required in the present invention, it is preferred as it can produce alias information which is more precise.

Blainey Col. 6 ll. 39-45. (emphasis added). This is precisely what Applicants have done, i.e., invented a "refinements as would occur to those of skill in the art." As claimed, Applicants' optimization techniques call for examining the uses made of an indirect reference to a variable made in an intermediate representation in source code. Depending on the uses (e.g., if no indirect stores are made to address taken variable) then a variable may be removed from the "address taken" alias set, and indirect references made to a variable may be replaced with direct references.

In response to Applicants' prior arguments, regarding independent claims 1, 11, 12, and 17, the Examiner asserts that *Archambault* discloses "replacing, if a particular use of the address is for an indirect reference to the variable, the indirect reference in the intermediate representation with a direct reference to the variable" as recited in each of the listed claims." For support, the Examiner cites to *Archambault* Col. 6 ll. 61-67: "Because the alias set for each use or de-reference of a local pointer (**indirect reference**) variable now contains . . . elements and is resolved. The alias sets computed by the front end of the compiler for intraprocedural analysis are replaced with the resolved alias sets (block 52) (**direct reference**)." (Final Office Action P. 7 (bold matter and ellipses added by Examiner)). The resolved alias set referred to in *Archambault* is just that: **an alias set**, albeit a more precise one. The resolved alias sets **are not** a direct reference to a variable. As described above, however, Applicants claim removing a variable from the address taken alias set entirely by removing indirect references to the particular variable and replacing the indirect reference with a direct one. Applicants respectfully submit that the Examiner is mistaking the resolved intermediate alias set references shown in *Archambault* at Col 6. ll. 6-61 with the complete removal of the indirect reference to a variable, along with the removal of the variable from an address taken alias set as claimed by Applicants. In other words, Applicant's invention would operate to remove certain alias sets (and replace indirect reference with direct ones) for certain variables listed in *Archambault* at Col 6. ll. 6-61.

Claims 7, 8, 23, and 24 stand rejected under 35 U.S.C. § 103(a) as being not patentable over *Blainey* (US 6,045,585) in view of *Archambault* (US 6,173,444), and further in view of *Lichtenstein* (US 6,077,311). In addition to the arguments presented above, related to *Archambualt* and *Blainey*, *Lichtenstein*, in combination therewith, fails to render applicants invention obvious.

Lichtenstein is directed to a method for marking a region of source code and creating an executable program that includes a copy of the marked code region that mimics the actions of the region from the larger program. It does so by adding data to the source code, and is in no way whatsoever related to optimizing techniques taught by Applicants' invention (or *Blainey* or *Archambault*, for that matter), or to optimizing techniques at all. Accordingly, it fails to teach, show, or suggest, in combination with *Blainey* and *Archambault*, or otherwise, optimizing an intermediate representation of source code based on the uses made of an indirect reference and address taken alias sets.

The secondary references made of record are noted. However, it is believed that the secondary references are no more pertinent to the Applicant's disclosure than the primary references cited in the Final Office Action. Therefore, Applicant believes that a detailed discussion of the secondary references is not necessary for a full and complete response to this Final Office Action.

Having addressed all issues set out in the office action, Applicant respectfully submits that the claims are in condition for allowance and respectfully request that the claims be allowed.

Respectfully submitted,

[illegible]

Gero G. McClellan
Registration No. 44,227
MOSER, PATTERSON & SHERIDAN, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Applicant(s)